COLLABORA

Embedded audio policies made easy with WirePlumber

George Kiagiadakis
Principal Software Engineer
george.kiagiadakis@collabora.com

Open First

What is a “policy”

- A set of rules
- From a security perspective: things allowed / disallowed
  - Access control
- In the audio system context:
  - Rules for routing audio streams towards devices
  - Rules for arbitration between streams
  - Rules for configuring streams & devices

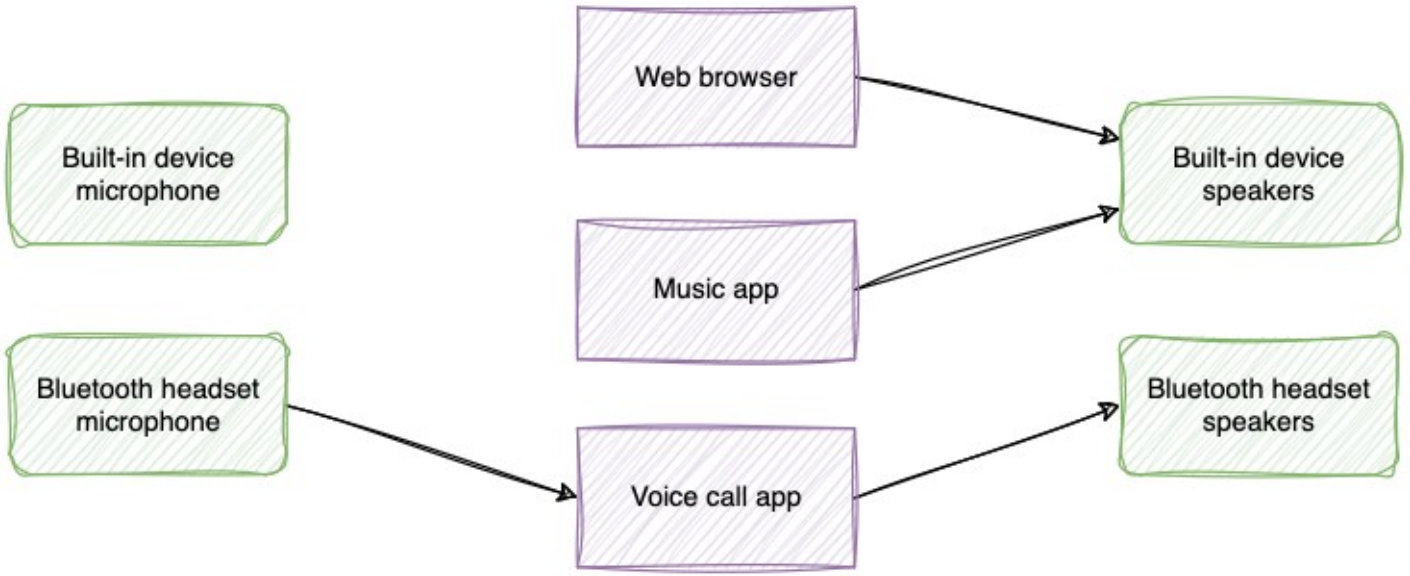
Embedded policy (??)

- **Use-case** based routing & arbitration
  - Also: “role-based” policy
- Phones, In-vehicle infotainment, In-flight entertainment, …
- Not all embedded systems need this kind of policy

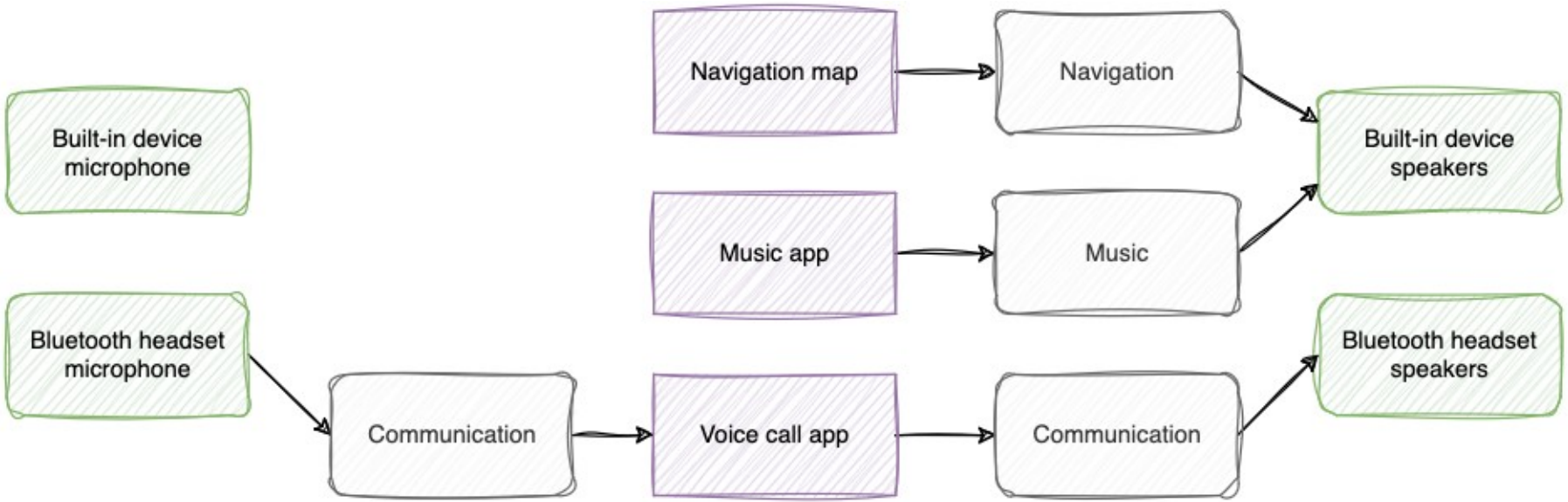
COLLABORA

Standard vs use-case based policies

Standard audio setup



Use-case based audio setup



COLLABORA

Use-case based policy components

- Associating application streams to use cases
- Associating use cases to hardware components
- Arbitrating between use cases

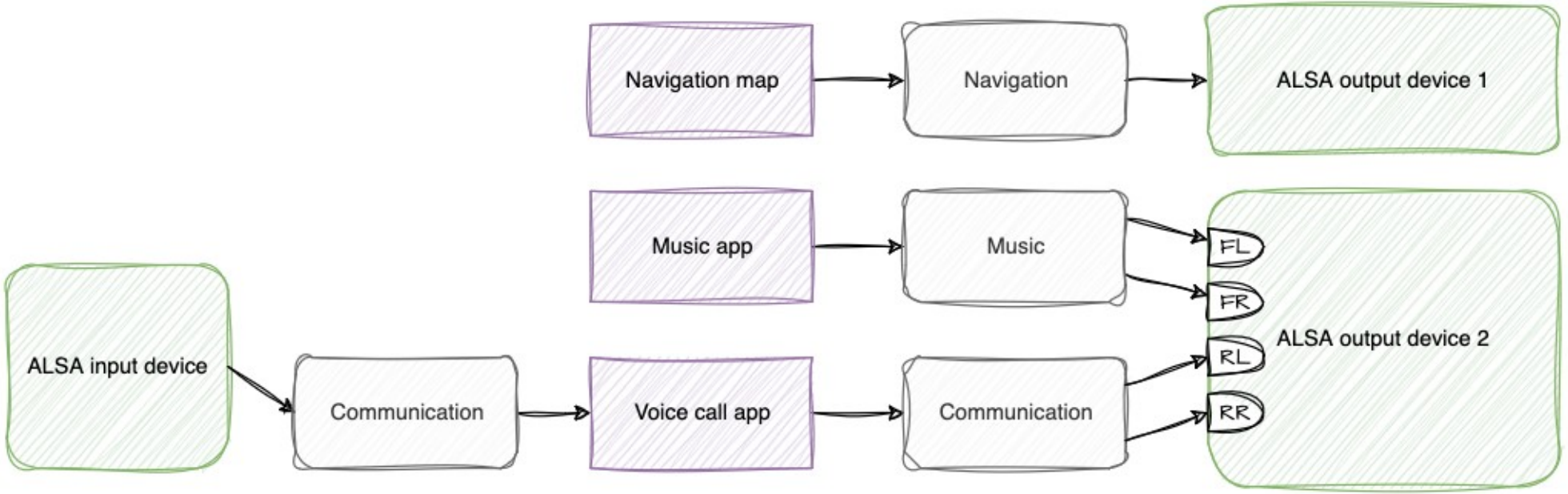
Streams → Use cases

- Music player app → “Music” use case
- WebRTC voice call app → “Communication” use case
- Navigation app → “Navigation” use case
- etc...

Use case → Hardware

- Which speakers / microphones to use for each use case?
  - Usually just 1 choice
- Much more common: Hardware DSPs
  - Each application stream is “forwarded” to the hardware DSP
    - Via separate sinks for each streams
    - Via specific channel assignments on a multi-channel sink

Forwarding to hardware DSP



Use case arbitration

- Navigation instruction while music is playing

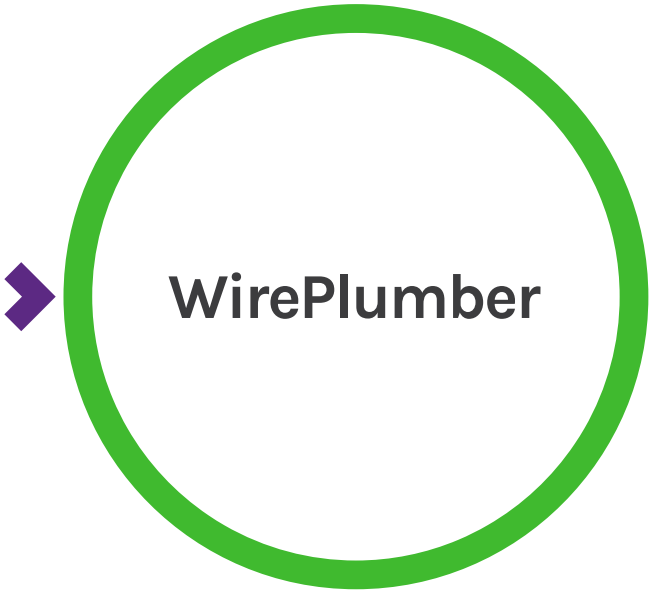
  → Duck music volume by 30%

- Important system announcement while video is playing

  → Pause video, ensure all paths except the announcement are muted

COLLABORA

Implementing with WirePlumber



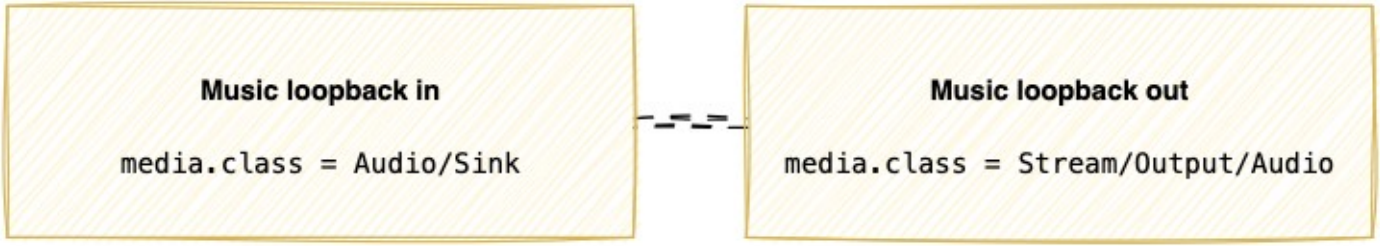
- Session manager (i.e. **orchestrator**) for PipeWire
- Desktop & embedded use cases
- Heavily modular / customizable
- Policy implemented via **Lua** scripts
- Event-based architecture

“Role-based” policy in WP 0.5

- Replaces the “endpoint” based policy in previous versions
- Allows using use-case based policy in combination with standard policy
  - With a setting to enforce role-based only, if necessary
  - Makes life easier in some scenarios, allows desktop use of some features
- Use cases implemented with simple loopbacks
  - Separate “app → use case” and “use case → hardware” routing policy

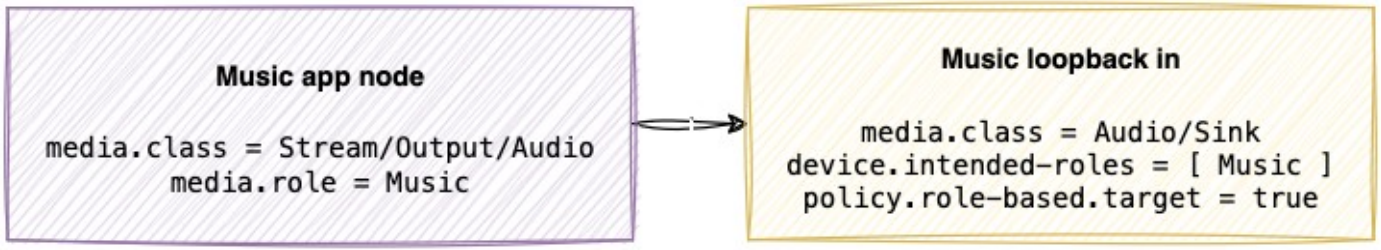
Loopbacks in PipeWire

- 2 coupled streams – input & output
  - One end registers as a target (“device”): “Audio/Sink” or “Audio/Source”
  - Other end registers as a regular stream: “Stream/Input/Audio” or “Stream/Output/Audio”



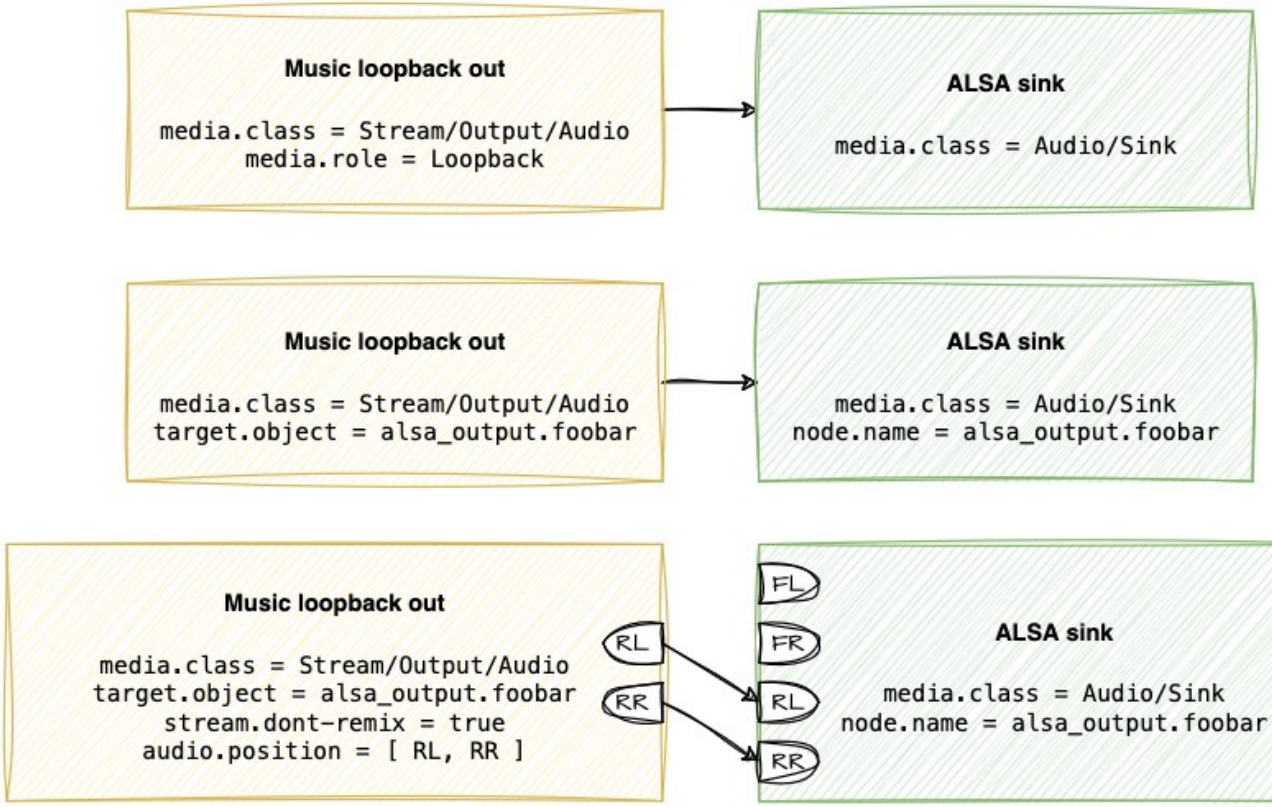
App streams → Use cases

- Select source / sink “target” node based on the application node’s “*media.role*” property
  - Setting exists to assign a default “media.role” for streams that don’t have this
  - PipeWire also has rules to apply properties such as “media.role” to streams
  - Scripting capability in WirePlumber for more advanced rules



Standard policy

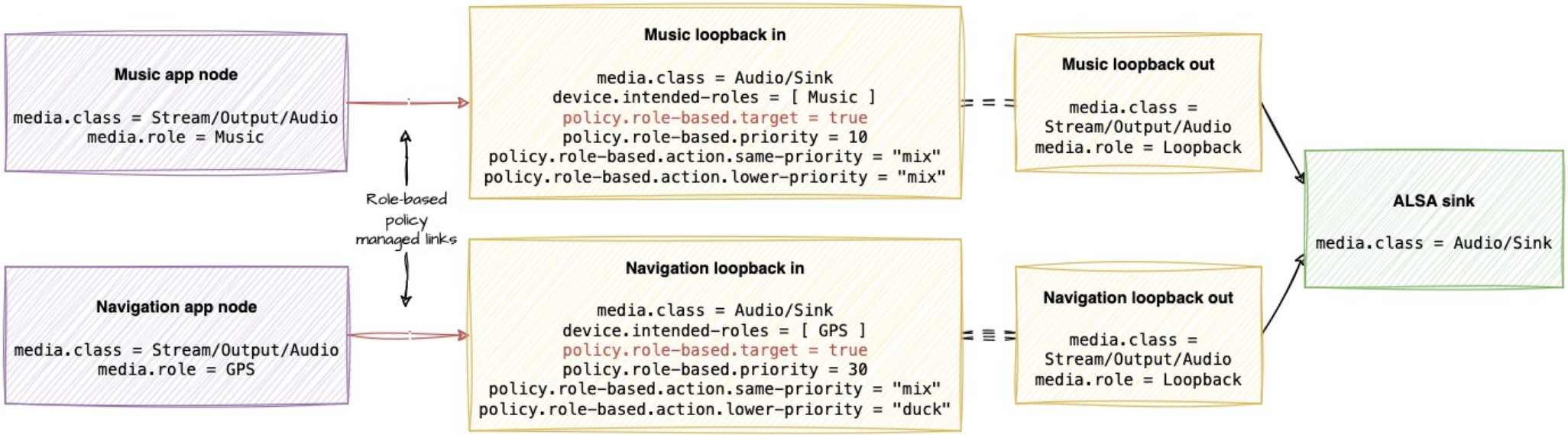
Use cases
→ Hardware



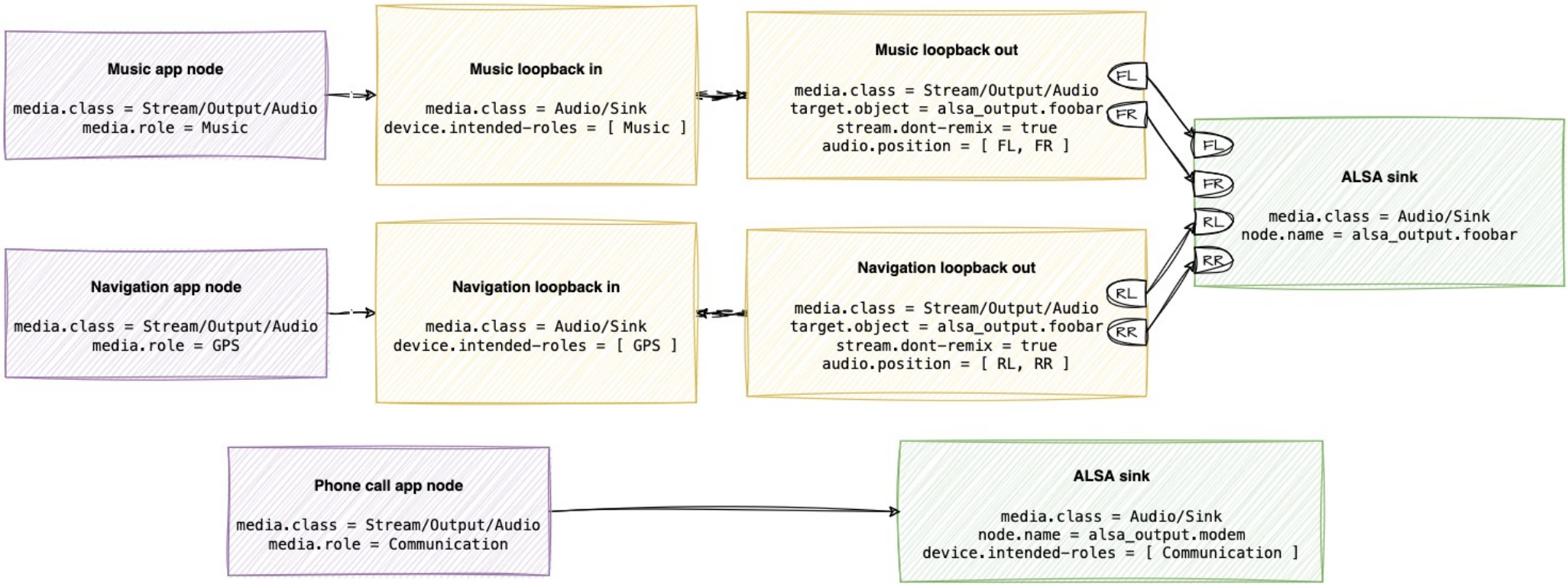
Arbitration

- Software based arbitration
  - Links to “*policy.role-based.target = true*” nodes are managed
  - Roles have priorities & actions: mix, cork, duck
    - Configured directly on “*policy.role-based.target*” node properties
  - Highest priority wins, other links follow designated action of the high prio role
- Hardware DSP based arbitration
  - No action taken, implementation dependent
  - May be combined with software “mix” & “cork” actions

Software based arbitration



Hardware DSP arbitration



Why loopbacks ?

- Implement software-based volume control
  - Useful for software based arbitration
- Remap multi-channel ALSA nodes to individual use case streams
  - Useful for hardware DSP based arbitration

Next steps

- Improve corking
  - Need to inform apps that they are being corked
  - Design exists, it’s a matter of implementation
- Hardware DSP management directly in WirePlumber
  - Implement communication to DSP with WP plugins
  - Requires collaboration - talk to me !

Thank you!